SCoRe: a Self-Organizing Multi-Agent System for Decision Making in Dynamic Software Developement Processes

Noelie Bonjean, Marie-Pierre Gleizes, Christine Maurel, Frédéric Migeon

SCoRe: a Self-Organizing Multi-Agent System for Decision Making in Dynamic Software Developement Processes

Noélie Bonjean[1], Marie-Pierre Gleizes[1], Christine Maurel[1], Frédéric Migeon[1]
[1]*Institut de Recherche en Informatique de Toulouse (IRIT),*
Université de Toulouse, France
Firstname.Name@irit.fr

Keywords: method fragments; adaptive multi-agent system; method process

Abstract: Software systems are becoming more and more complex. A common dilemma faced by software engineers in building complex systems is the lack of method adaptability. However, existing agent-based methodologies and tools are developed for particular system and are not tailored for new problems. This paper proposes a new tool based on SME for self-constructing customized method processes. Our approach is based on two pillars: the process fragment and the MAS meta-model. These two elements are both defined and considered under a specific agent-oriented perspective thus creating a peculiar approach. Our work is based on the self-organization of agents, making it especially suited to deal with highly dynamic systems such as the design of an interactive and adaptive software engineering process.

1 Introduction

In the Multi-Agent community, it is now obvious to consider that Multi-Agent Systems (MAS) are a relevant means to design complex systems. Their features fit well the openness, the big number of entities and of interactions and the non linearity which characterize complex systems. The MAS community has been prolific to define software engineering methods (Bergenti et al., 2004; Henderson-Sellers and Giorgini, 2005) in order to guide designers with respect to the wide range of MAS properties. Facing the numerous methods, a development team needs help to choose and to execute the relevant process according to the development context which is defined by the System Under Study (SUS) characteristics as well as the team capabilities and preferences. Moreover, this process may need to combine advantages of several other ones (for example, requirements analysis in TROPOS method and self-adaptation in ADELFE method (Morandini et al., 2009)). It also may need adaptation during design time to reflect SUS and team instability. Therefore, our goal is to provide new tools for designing complex systems where the method must be adapted to the development context.

Coming from Situational Method Engineering research (Henderson-Sellers and Ralyté, 2010), the aim of decomposing processes into pieces is to adapt the process to the characteristics of the business problem and to the level of expertise of engineer teams (Ralyté, 2004). A process can then be defined by assembling the pieces of methods, called fragments, in order to suit the context (the situation) changes. The Agent-Oriented Software Engineering (AOSE) community contributed to this research splitting up methods into fragments and providing precise descriptions of them (ADELFE, INGENIAS, PASSI, SODA, TROPOS...). This kind of work is mainly done in the Foundation for Intelligent Physical Agents[1] (FIPA) context.

Currently some propositions to combine fragments have been already made, but they are mainly based on the know-how of the method engineers. This is the case with the ProDe (a Process for the Design of Design Processes) (Seidita et al., 2010) approach. In this paper, we propose an automatic way to design a method process based on MAS technology. The process is constructed by combining fragments ”on the fly” to self-adapt to the specific situation of the project.

We propose an original system called SCoRe (Self-Combined method fRagments) to automatically build a self-adaptive design process where each fragment is encapsulated in an autonomous agent. It relies on the self-organization of its agents, making this approach especially suited to deal with highly dynamic systems such as the design of an interactive and adaptive Software Engineering Process (SEP)

[1] http://www.fipa.org

(Bernon et al., 2005).

This paper is organized as follows. First, section 2 explains the aim of this system. Then, section 3 introduces the system of SCoRe and details the behavior of the involved agents as well as their interactions. Section 4 focuses on tests and explains the results obtained. Finally, section 5 describes related works before concluding in section 6.

2 Requirements and Characteristics of SCoRe

The contribution of the work explained in this paper, lies in the self-adaptive multi-agent system implementation for self-composition and self-organization of method fragments. This section presents the challenges.

2.1 Adaptation

While the demand for specific, complex and varied system continues to grow, current methods in the MAS domain remain limited and sometimes not well adapted. For example, in order to propose a simulation-based process for the development of MASs which incorporates a simulation phase for the prototyping of the MAS being developed and for functional and non-functional validation, PASSIM (Cossentino et al., 2008) was obtained by integrating method fragments from PASSI for carrying out the analysis, design and coding phases, and the Distilled State Charts (DSC)-based simulation method for supporting the simulation phase. The need for well-defined guidelines that will make the development process more efficient and more effective has become crucial. Currently, there is no single methodology that can be uniquely pointed as ”the best”. Until now method adjustments to the specific requirements and constraints are mixed in ”local” adaptations and modifications of existing one. In order to succeed in creating good situational methods, i.e., methods that best fit given situations, fragment representation and cataloguing are very important activities. In particular, the fragments (sometimes addressed as process fragments, method fragments or chunks) have to be represented in an uniform way that includes all the necessary information that may influence their retrieval and assembling.

2.2 Fragment Standardisation

Method fragments are first identified by examining existing methods. These method fragments are made according to templates defined by repository designers. Therefore the choice of fragments granularity relies on designers. According to the Rational Unified Process, the methods are defined following different levels of granularity: phase, activity and step. The granularity issue of these method fragments presents important challenges. The ”step” level involves a specific and fiddly task but also requires perfect knowledge of methods and long work. This fragmentation is very fine-grained and provides a greater number of fragments. For instance, in ADELFE, the analysis phase is composed of four activities, the first of which is *Analysis of domain*. *Analysis of domain* consists in two steps: *Identify the active and passive entities* and *Study interactions between the entities*. These steps are related and interdependent. This low level of granularity is therefore useless and inaccurate in this situation. On the other hand, the ”phase” level of granularity could form huge complete fragments. The coarse-grained granularity promotes the redundancy issue. The duplication of activities or steps may occur with high granularity. By consequences, an activity or step may be included in different fragments. The risk that this happens grows up with the level of granularity. In addition, the assembling possibilities are therefore minimized.

2.3 Complexity

Currently, ten AOSE methods are fragmented, each one composed of approximately twenty fragments. Such fragments constitute the root constructs of the method itself and they have been extracted by considering a precise granularity criterion: each group of activities (composing the fragment) should significantly contribute to the production/refinement of one of the main artefacts of the method (for instance, a diagram or a set of diagrams of the same type). Following this assumption, fragments obtained from different methods are based on a similar level of granularity.

Besides, to design a process manually means studying for the compatibility of each fragment with the others i.e. approximately twenty thousand possible combinations. Although this number can be decreased by the knowledge and the know-how of process engineers, the work remains long and irksome. It is why we propose to design the system SCoRe to self-combine and self-organize fragments.

2.4 User Requirements

In our approach, a new complete process is self-designed contingent on a situation. A complete process enables engineers to visualise all activities and to

have a whole view of the process. SCoRe focuses on adaptation during process execution. At every step, the development team is advised by SCoRe on its next fragment choice according to the running features. If the features evolve, this advice may therefore differ from the following fragment initially suggested.

The studied solution is based on the fragments agentification in order to self-design an adaptive process. This choice is justified by the problem complexity which is mainly due to the huge number of fragments. Indeed, a complex system cannot currently be designed without bug caused by designers. Assisting the designer during the system utilization would reduce the number of bugs and make the system most suitable to the current situation. The adaptation is therefore required. As for components assembling, the fragments combination needs features. In our approach, they correspond to MAS Metamodel Elements (MMME). Two fragments are therefore combined if one produces the MMME required by another.

3 Combining Method Fragments with an Adaptive Multi-Agent System

The general structure of the Self-Combined method fRagments (SCoRe) proposed is described in this section, before detailing the behaviors and the interactions of the agents composing it.

3.1 SCoRe Components

We consider a method process as a set of assembled method fragments which are linked through their own required or produced MMMEs. Establishing a method process consists in combining some of the fragments taking into account the user-defined objectives and knowledge.

The main goal of SCoRe is to suggest a tailored process. For that, SCoRe learns the context to apply on fragments, in order to sustain this evolution. SCoRe acts without relying on a model of the processes, meaning that it is only able to take into account the users' knowledge and objectives, and to observe the evolution of the running process on which MMMEs are available, in order to decide on the fragments to add. The best possible running process is therefore designing according to a context.

Therefore, to build a process, we mainly need to know the objective and the knowledge of the user (designer) and the context.

3.1.1 Users' Objectives and Knowledge

In order to design a tailored process, SCoRe requires some information about the wanted system and about the users. Actually, these informations enable to select fragments which make up the suggested process. On the one hand, the user has to give his/her knowledge about the known technologies, methods and paradigms. Figure 1 shows an example of characteristics and we do not provide here the exhaustive list of characteristics. On the other hand, the user has to describe briefly, in a given form provided in figure 2, the intended system by defining the field of application, the phase corresponding to the initial and final work product and the type of system.

Users' Characteristics		
Technologies	Methods	Paradigms
☐ UML	☐ ADELFE	☐ Agent
☐ Java	☐ PASSI	☐ Cooperation
☐ SpeADL	☐ INGENIAS	☐ Emergence
☐ MAY	☐ TROPOS	

Figure 1: Example of users' characteristics

System Characteristics			
Field	Phase to Initial Work Product	Phase to Final Work Product	Type of System
☐ Automotive	☐ Analysis	☐ Analysis	☐ Profiling
☐ Biology	☐ Requirement	☐ Requirement	☐ Simulation
☐ Maritim Surveillance	☐ Implementation	☐ Implementation	☐ Self-regulation
☐ Aviation Industry	☐ Design	☐ Design	☐ Optimization
			☐ Manufacturing Control

Figure 2: Example of system characteristics

3.1.2 Context

The context is a set of elements external to the activity of an entity. It describes the environment wherein the entity evolves. Moreover, the context has an influence on the process of fragments selection.

In processes under construction, the context is made up of users' objectives and knowledge, available fragments and the elements included in the running process.

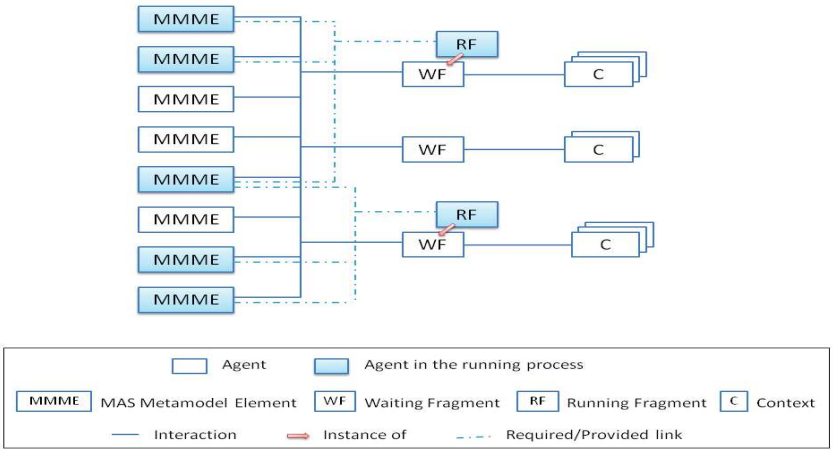

Figure 3: Example of agents and their relationships in SCoRe

3.1.3 Agents

SCoRe is composed of four distinct kinds of agents, following a perception-decision-action lifecycle, which cooperate according to the adaptive multi-agent systems theory described in (Capera et al., 2003). The basic idea underlying this cooperation consists, for every agent in an adaptive multi-agent system, in always trying to help the agent which encounters the most critical situation from its own point of view.

Figure 3 gives the structure of SCoRe designing a method process. The different types of agents involved are shown, as well as the links modelling the existing interactions between them. Actually, SCoRe is made up of the following agents:

- **MAS Metamodel Element (MMME)**: required or produced by a Running Fragment, its aim is to choose which fragment it will be linked to. A MMME is connected with all the waiting fragment and the Running Fragments which are included in the running process and which produce or consume it.
- **Waiting Fragment (WF)**: its purpose is to integrate its instances (running fragment) in a process once it is in an adequate situation. They are linked to the MMMEs and a set of context.
- **Running Fragment (RF)**: it aims at finding its localization inside the running process. A RF is linked with the MMMEs encompassed in the running process that it produces or requires.
- **Context (C)**: related to a fragment, it aims at evaluating its relevance according to the MMMEs already involved in the running process and the users' objective and knowledge. The context agent is related to a fragment for which it evaluates its relevance to be added in the running process.

3.2 General Behavior of SCoRe

3.2.1 Prerequisite

Some prerequisites are necessary for the execution of SCoRe. Actually, in order to help the fragments selection in SCoRe, the user has to fill in two forms about its objectives and its knowledge. Firstly, the user completes its knowledge that will be mainly used to select fragments. Secondly, the user characterizes the wished system. From the ticked fields, the initial and final MMMEs are extracted. Actually, according to user's knowledge and the initial and final phase selected, MMMEs are included in the running process by SCoRe.

3.2.2 Starting of the Running Process Construction

When the user provides the prerequisite items, Waiting Fragment agents and their Context agents are created. They know each other however the Waiting Fragment agents don't know the others Waiting Fragment, and their associated Context agents don't know each other.

Besides, the MMMEs corresponding to users' problem are created with the acquaintance of the available Waiting Fragments and they are then included in the running process. The construction of the running process starts therefore with the MMMEs. These MMMEs are said *initial* and *final*. While a MMME aims at being linked to at least two fragments, i.e. one which produces it and one which con-

sumes it, an *initial*, respectively *final*, MMME aims at being linked to at least one fragment which consumes it, respectively produces it.

The *initial* and *final* MMME start the running process construction by interacting with all the waiting fragments.

3.2.3 Running Process Construction

As it is shown in figure 3, an agent communicates to some others agents. The first agent interactions correspond to the *initial* and *final* MMME looking for a fragment. When a waiting fragment receives a message from a MMME agent, it solicits their own contexts. The context self-evaluates itself. Then it answers by giving its relevance to the waiting fragment which sends it to the MMME in turn. According to the different answers, one of the requesting MMME selects a waiting fragment. The selected waiting fragment is then ready to create a running fragment. The created running fragment is added in the running process. Being related to input and output MMMEs, when a running fragment is added in the running process, it links itself to the MMMEs already present in the running process. If one of its MMMEs is missing, the running fragment creates it. Then, the created MMME agent is added in the running process. The MMME agents request the waiting fragment agents until to be satisfied.

Next sections will provide a more detailed description of these agents behaviors and interactions.

3.3 Behavior of the Agents

The behavior of an agent is a life cycle consisting of the sequence: (i) perception of the environment (including communication aspects), (ii) decision that allows it to identify the state in which it lies and actions to be performed, (iii) execution of decided actions. The life cycle starts when the agent is created and completes when the agent dies.

Besides, an agent that intervenes in an AMAS is composed of different parts that produce its behavior: skills, aptitudes, the interaction language, world representations, Non Cooperative Situations, and criticality and/or confidence.

For an agent, criticality represents the degree of non-satisfaction of its own goal. It enables an agent to determine the relative difficulty of agents in its neighborhood. Evaluation methods and calculation of the criticality are specific to each type of agent. The confidence of an agent is an internal measure that provides information on the reliability of the decision on actions intended.

These two notions guide the behavior of the SCoRe agents and will be presented in the following subsections.

3.3.1 The MMME Agents

The MMME agents represent the links between running fragment agents. Their individual goal is to be incorporated in the running process. The MMMEs behavior is represented by an automaton with two states: *non incorporated* and *incorporated*. The *non incorporated* state corresponds to a MMME linked to at least one running fragment which respectively produces or consumes it. In this state, it requests new fragments (respectively a consumer or a producer). It is looking for a relevant fragment to which it can be linked. Actually, it requests all waiting fragments. The relevant waiting fragments answer it by giving their confidence. The confidence of the waiting fragment provides information on its reliability proposition. The most relevant fragment will be chosen by the MMME agent for being a potential fragment to be added in the running process and the MMME updates its confidence with the maximum of the relevant fragment confidence. Let $Cf_F = \{Cf_1 ... Cf_n\}$ be a set of the relevant fragments confidence and Cf_e the confidence of the MMME e, then:

$$Cf_e = Max(Cf_F)$$

Furthermore, these agents are able to evaluate their own criticality. It is a ratio of the number of waiting fragment answers over its number of executed lifecycle.

For the MMME e, let A_e be its number of waiting fragments answer, L_e be its number of executed lifecycle and Cr_e be its criticality, then:

$$Cr_e = A_e / L_e$$

Therefore, the MMME agents cooperate to choose the most relevant fragment among the ones suggested according to their own criticality.

The *incorporated* state is reached when the MMME agent is linked with at least two fragments: one consumer and one producer. The initial or final MMMEs provided by the designer have only to be linked respectively to at least one producer and one consumer.

3.3.2 The Waiting Fragment Agents

Each waiting fragment agent has an associated set of context agents. Their goal is to be integrated in a process once it is in an adequate situation. For that, when a waiting fragment agent receives requests from

MMMEs which are looking for a fragment, it forwards the request to its context agents, if the waiting fragment agent considers itself as a potential solution. A waiting fragment agent considers itself to be solution if the requesting MMME belongs to its own required or provided MMME. Then, the waiting fragment agent waits the answer from its context agents. It updates its confidence and sends it to the MMME. Its confidence is calculated in adding the confidence received from the context. Should the opposite occurs, the waiting fragment agent sends an answer to MMME with no relevance. Let $Cf_C = \{Cf_1 ... Cf_n\}$ be a set of the context confidence and Cf_f the confidence of the waiting fragment f, then:

$$Cf_f = \sum_{i=0}^{n} Cf_i$$

Moreover, a waiting fragment agent can be selected to be inserted in the running process. Actually, when the waiting fragment agent receives a message from the MMME to inform it that it is selected, the waiting fragment agent creates a running fragment agent. Moreover, it sends the information to its context agents as a feedback.

3.3.3 The Running Fragment Agents

A running fragment agent is created by a waiting fragment agent which represents it in the running process. It is introduced on time in the process. Its aim is to be incorporated in the running process. Its behavior changes according to its current state and its perception. The current state of a running fragment agent corresponds to *non incorporated* and *incorporated*. Actually, a running fragment agent is said *incorporated* when all the required MMMEs are in the *incorporated* state and at least one of the provided MMMEs is *incorporated*. Otherwise its state is *non incorporated* and the running fragment agent makes links with each MMME agent existing in the running process on which a link is physically possible. Moreover, when the running fragment agent is inserted in the running process, it adds the required or produced MMMEs which are missing in the running process. Furthermore, these agents are able to evaluate their own criticality. It is calculated from the criticality of required or produced MMME(s) and their current state.

Let $Cr_C = \{Cr_1 ... Cr_n\}$ be a set of the provided or required MMME criticality and Cr_f the criticality of the running fragment f, then:

$$Cr_f = \sum_{i=0}^{n} Cf_i$$

3.3.4 The Context Agents

The goal of the context agents is to represent a situation leading to a specific method process. They do not aim to model what is happening inside the system, but rather aim at selecting the fragment to add in the current situation to reach the objectives. When such an agent finds itself in its triggering situations, it notifies the waiting fragment agent, by submitting its confidence according to its own knowledge.

In order to know when the fragment is relevant, a context agent relies on two different sets of information. First, a collection of input values represents the set of user and system characteristics. This element enables the context agent to know if it has to be triggered or not. Besides, a context agent possesses a set of forecasts, which describes the impact of the action proposed on the satisfaction of the both user and system characteristics. Moreover, a context agent possesses a set of metrics, which describes the impact of the action proposed on the running process (Bonjean et al., 2012). Those input values are modified during the life of a context agent. According to its behavior, from different feedback that it receives, a context agent adjusts its confidence, a value representing its relevancy to add the waiting fragment to which it is linked.

Finally, the behavior of a context agent is represented by an automaton. Each state relates its current role in the MAS. A total of three different states exist: *disabled*, *enabled* and *selected*. The context agent can switch from a state to another thanks to the messages it receives from other agents in the system. A disabled context agent considers itself non-relevant in this specific situation. An enabled context agent thinks that it is relevant and potentially deserves to be selected. It then computes its confidence and sends it to the corresponding waiting fragment agent. Finally, a selected context agent is validated by a waiting fragment agent and its associated fragment is added in the running process. This selected context agent has then to observe the consequences of its actions in order to reinforce or to update its confidence.

Let f be the fragment linked to the context x, let $P_f = \{p_1 ... p_m\}$ be a set of its characteristics and E_f be the proportion of its required or produced MMMEs already included in the running process.

Let $P_u = \{p_1 ... p_n\}$ be a set of users' characteristics and U_f the proportion of matching between to the fragment and users characteristics.

$$U_f = \frac{P_f \cap P_u}{P_f}$$

Let Cr_x be the criticality and Co_x the confidence of the context x, we calculate:

$$Cr_x = (U_f + E_f) * Co_x$$

4 Results and Analysis

The test coverage we wanted to obtain is about correctness and adaptability. The correctness of a method is asserted when it is said that the method is correct with respect to a specification while the adaptability is asserted when a method adapts itself efficiently and fast to changed circumstances. Before presenting the test cases that we defined, we discuss of the specificity of method engineering.

The designed method is conceived of not a single interdependent entity but as a set of disparate fragments. Therefore, in order to show the correctness of a new method process, the method has to be evaluated by method engineers. There are two ways of evaluating a method, which can be complementary. The first one is based on empirical studies that have been conducted by many practitioners and researchers. The second one can be obtained more automatically from metrics or impartial indicators (Bonjean et al., 2012). This kind of experience is complex and can take a long time to obtain sufficient results. As a consequence, we firstly focus on the functional adequacy and the dynamic adaptation to specific situation.

The first test has been carried out with known method processes to show the correctness of SCoRe. We show that Score enables to compose a known method back from its fragments. Concerning adaptability test, we conducted them with fictive processes. Combining known methods are very complex task. Until now, the inter-operability and semantic matching of fragments from different known methods stay an important challenge. Actually, in this problem, some works base on standardisation of fragments notion and of their description. For this reason, we simulate methods. Therefore we used fictive processes which have been simplified. Actually, they are defined in the following way. They are composed by four fragments while a known method is made up of approximately twenty fragments. These tests show how SCoRe is able to adapt itself: on the one hand, how SCoRe adapts itself to a context and on the other hand, how SCoRe adapts itself to design a new tailored method process.

4.1 Without Users' Characteristics

Preliminary results show the correctness of SCoRe. The first test aims at verifying that the system self-designs and proposes a complete method process. In this case, at the set-up, all fragments from a repository obtained from current methods such as ADELFE[2] INGENIAS (Pavòn et al., 2005) TROPOS (Bresciani et al., 2004) and PASSI (Cossentino et al., 2006) are provided without order. The test is validated if SCoRe is able to combine a predefined process as entire ADELFE or PASSI process. In order to have this result, we have tagged the initial and final MMME to force SCoRe to choose the corresponding fragments. During the execution, the SCoRe agents interact and the expected process is built up again. SCoRe enables to design a known process.

We execute two different tests :

1. the initial MMME correspond to *ADELFE problem* and the final MMME is the *Cooperative Agent Behavior Code*. After the execution of SCoRe, the ADELFE process is proposed.
2. the initial MMME correspond to *PASSI problem* and the final MMME is the *PASSI Code*. After the execution of SCoRe, the PASSI process is designed.

Thus, the test shows the accuracy of the agents behavior. SCoRe is therefore able to propose a complete process.

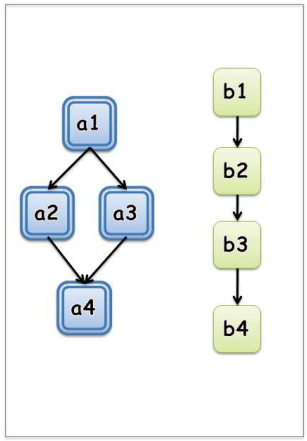

Figure 4: Test With Hand-made Users' Characteristics

4.2 With Users' Characteristics

From now on, in these tests, we simulate two short fictive processes. These processes named A and B are defined to illustrate the test results. A is broken into four fragments a1, a2, a3, a4 where all fragments are sequential except for a2 and a3 which are alternative. The process B is broken in four sequential fragments

[2] ftp://ftp.irit.fr/IRIT/SMAC/DOCUMENTS/RAPPORTS/

b1, b2, b3 and b4. The processes are depicted in figure 4.

The previous test showing the capability of SCoRe to design a complete process, the following tests extend it by taking into account the users' characteristics during the fragments selection. The users' characteristics are integrated iteratively. Actually, they are firstly used to select a process and they are secondly used to select tailored fragments composing the process.

4.2.1 With Hand-made Users' Characteristics

This test extends the first one, by using hand-made users' characteristics as in figure 1. In this case, the aim of the experiments is to check if SCoRe design a method process corresponding to the users' characteristics. The input values corresponding to a pool of users' characteristics of each context are defined before the simulation, and remain unmodified during its run. The context agent submits a higher confidence when its knowledge matches with the users' characteristics. In any situation, we controls that the system advises the most adapted process. In this case, the waiting fragment agents represent only the fragments from the independent fictive processes A and B. We simulate both processes with the initial MMME: *user's problem* and the process A provides the final MMME: *Product A* while the process B provides the final MMME: *Product B*.

Figure 4 shows the obtained processes during this test. We execute three different tests :

1. the user knows the method A and its technologies and paradigms. From the user's objective and his characteristics, the initial MMME correspond to *user's problem* and the final MMME is the *Product A*. After the execution of SCoRe, the method process A is designed (see the bleu process in figure 4).
2. the user knows the method B and its technologies and paradigms. From the user's objective and his characteristics, the initial MMME correspond to *user's problem* and the final MMME is the *Product B*. After the execution of SCoRe, the method process B is designed (see the green process in figure 4).
3. the user knows both method A and B and their technologies and paradigms. After the execution of SCoRe, one of the two method process is designed (see figure 4).

As a result, one method process is chosen as the most adapted for the specified situation. Score enables therefore to adapt a proposed process to the users' characteristics.

4.2.2 With Adaptive Users' Characteristics

This test extends the previous one by implementing the adaptive behavior of the agents. It aims to combine some processes. In this case, fragments from different processes are assembled in order to obtain a new process more adapted. The initial and final MMMEs are common to the both processes. Moreover, required MMMEs can be provided by a fragment from another process. We supposed that the provided fragments from A and B are compatible with each other and a part of them with the user's capabilities. Actually, figure 5 shows the existing dependency between the fragments from the both processes.

In this test, according to users' characteristics, SCoRe is able to produce a new process based on fragments from both initial processes in addition to processes already known. Actually, figure 5 shows an example of obtained result where the new process (the bleu one) is composed of a1, b2, a3 and b4. According to users' characteristics, this process is advised as the most accurate.

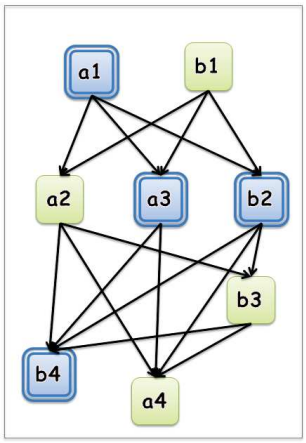

Figure 5: Test With Hand-made Users' Characteristics

5 Related Works

In this paper, we have presented a new approach for modelling and self-composing method fragments. Apart from application field, several recent works exploit the lessons of adaptive self-organizing natural and social system to enforce self-awareness, self-adaptability, and self-management features in distributed system. We mention below only the works closest to our approach. The most strictly related approaches are concerned with the problem of dynam-

ically composing and adapting fragments or components.

Some approaches of components agentification have been developed. They aim at allocate components to agent properties such as autonomy and interaction. (Kuikka and teknillinen tutkimuskeskus, 1999) proposes a pattern of components agentification which aims at integrating components. Another approach uses an extension of ContractNet protocol for finding components in libraries (Hara et al., 2000). A function which deals with request as message is added in the components stored in libraries as an agent. The agents have knowledge about specificity of the component and its ability to answer a need. Our proposition has the same goal with a view to distribute not only fragments research but also their adaptation and their composition. This allocation of reuse process enables several assembling strategies for instance.

Besides, Web Services represent today's reference standard technology for the set up of distributed systems that need to support machine-to-machine interaction among heterogeneous applications distributed over a network. The automatic composition and adaptation of services has been explored using a variety of AI planing engines. A review of further automated service composition methods may be found in (Rao and Su, 2005). In (Thomas et al., 2009), a set of workflow fragments are composed in ad hoc wireless mobile environments. A workflow fragment is a set of tasks linked together with conditions. Actually each task has preconditions (input) and postconditions (output) related to this execution. Two workflow fragments are linked if the precondition of one is the same as the postcondition of the other one. This approach aims at designing dynamic construction of custom, context-specific workflows in response to unpredictable and evolving circumstances by exploiting the knowledge and services available within a given context. For that, a graph made up of all workflow fragments is built up before exploring and pruning it. The composition issue is simplified. Actually, on the one hand, the workflow is defined as a directed acyclic graph with vertices denoting tasks and edges defining an execution order along with the flow of data and control and, on the other hand, the conditions have a unique name in a workflow fragment. As presented approaches, ours is based on current data base of fragments and on MAS metamodel elements as input and output. It proposes an automation of method fragments composition. The way to integrate the method fragment in the process is different. Actually, in running development, our approach can take into account process adaptation according to development context.

6 Conclusion and Future Works

This paper presentes SCoRe, an adaptive multi-agent system, which designs a tailored process by combining fragments together. Each agent composing the adaptive multi-agent systems follows a local and cooperative behavior, driven by the use of their confidence. The four different kinds of agents, composing the SCoRe system, were defined in order to self-design and self-combine a tailored method process without relying on the method engineer. The resulting behavior of the SCoRe system is the ability to design process and adjust the proposed process according to the characteristics of application domain and users profile. This first prototype allowed to enhance our experience on practical problems such as metamodel compatibility, parameters composition or fragments adaptation to specific field.

However, there is still room from improvements in some aspects of this approach. For example, the inter operability and the semantic matching of fragments from different methods are still missing. In this problem, some works axis on standardisation of fragments notion and of their description. The metamodel definition or ontologies for software process could be used. Another approach from model-driven engineering is the Model Transformation By Example (Kappel et al., 2012). The concept is to make easier model transformation writing without generic model in favour of requested generated transformation. Thus fragments drawing on similar metamodels could be made up automatically.

Moreover, another important point is the evaluation of the designed process. Actually, despite the proposal of elaborate new tailored method processes, methods are built intuitively by adopting some fragments from different methods. It is therefore difficult to evaluate and compare methods. In order to made a right choice, it is necessary to evaluate the method obtained with SCoRe.

Finally, this SCoRe system will be confronted to real users' problems with known method fragments, in order to allow its comparison with existing methods.

REFERENCES

Bergenti, F., Gleizes, M., and Zambonelli, F. (2004). *Methodologies And Software Engineering For Agent Systems: The Agent-oriented Software Engineering Handbook*. Kluwer Academic Pub.

Bernon, C., Camps, V., Gleizes, M.-P., and Picard, G. (2005). Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology . In Henderson-Sellers,

B. and Giorgini, P., editors, *Agent-Oriented Methodologies* , volume ISBN1-59140-581-5, pages 172–202. Idea Group Pub, NY, USA.

Bonjean, N., Chella, A., Cossentino, M., Gleizes, M.-P., Migeon, F., and Seidita, V. (2012). Metamodel-Based Metrics for Agent-Oriented Methodologies (regular paper). In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), Valencia, 04/06/2012-08/06/2012*.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agent and Multi-Agent Systems (8)*, 3:203–236.

Capera, D., Georgé, J.-P., Gleizes, M.-P., and Glize, P. (2003). The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents. In *International Workshop on Theory And Practice of Open Computational Systems (TAPOCS at IEEE 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2003) (TAPOCS), Linz, Austria, 09/06/2003-11/06/2003*, pages 389–394, http://www.computer.org. IEEE Computer Society. Pages de la publication : –.

Cossentino, M., Fortino, G., Garro, A., Mascillaro, S., and Russo, W. (2008). Passim: A simulation-based process for the development of multi-agent systems. *International Journal on Agent Oriented Software Engineering (IJAOSE)*.

Cossentino, M., Sabatucci, L., and Seidita, V. (2006). Method fragments from the passi process. Technical Report RT-ICAR-21-03, Istituto di Calcolo e Reti ad Alte Prestazioni - Consiglio Nazionale delle Ricerche.

Hara, H., Fujita, S., and Sugawara, K. (2000). Reusable software components based on an agent model. In *Proceedings of the Seventh International Conference on Parallel and Distributed Systems: Workshops*, ICPADS '00, pages 447–, Washington, DC, USA. IEEE Computer Society.

Henderson-Sellers, B. and Giorgini, P. (2005). *Agent-oriented methodologies*. Information Science Reference.

Henderson-Sellers, B. and Ralyté, J. (2010). Situational method engineering: State-of-the-art review. *J. UCS*, 16(3):424–478.

Kappel, G., Langer, P., Retschitzegger, W., Schwinger, W., and Wimmer, M. (2012). Model transformation by-example: A survey of the first wave. In Dsterhft, A., Klettke, M., and Schewe, K.-D., editors, *Conceptual Modelling and Its Theoretical Foundations*, volume 7260 of *Lecture Notes in Computer Science*, pages 197–215. Springer Berlin / Heidelberg. 10.1007/978-3-642-28279-9_15.

Kuikka, S. and teknillinen tutkimuskeskus, V. (1999). *A Batch Process Management Framework: Domain-specific, Design Pattern and Software Component Based Approach*. VTT publications. Technical Research Centre of Finland.

Morandini, M., Migeon, F., Gleizes, M.-P., Maurel, C., Penserini, L., and Perini, A. (2009). A goal-oriented approach for modelling self-organising mas. In Aldewereld, H., Dignum, V., and Picard, G., editors, *Engineering Societies in the Agents World X*, volume 5881 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin / Heidelberg.

Pavòn, J., Gòmez-Sanz, J. J., and Fuentes, R. (2005). The ingenias methodology and tools. In *Agent Oriented Methodologies*, chapter IX, pages 236–276. Henderson-Sellers.

Ralyté, J. (2004). Towards situational methods for information systems development: Engineering reusable method chunks. *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education*, pages 271–282.

Rao, J. and Su, X. (2005). A Survey of Automated Web Service Composition Methods. In *LNCS*, volume 3387/2005, pages 43–54. Springer.

Seidita, V., Cossentino, M., Galland, S., Gaud, N., Hilaire, V., Koukam, A., and Gaglio, S. (2010). The metamodel: a starting point for design processes construction. *International Journal of Software Engineering and Knowledge Engineering*, 20(4):575–608.

Thomas, L., Wilson, J., Roman, G.-C., and Gill, C. (2009). Achieving coordination through dynamic construction of open workflows. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '09, pages 14:1–14:20, New York, NY, USA. Springer-Verlag New York, Inc.